Internet-Draft
Expires: October 20, 2009
Intended Status: Proposed
Filename: draft-keromytis-keynote-x509-02.txt

A.D. Keromytis
Columbia University
March 25, 2009

X.509 Key and Signature Encoding for the KeyNote Trust Management System

Status of this Memo

Copyright and License Notice

Abstract

This memo describes X.509 key identifiers and signature encoding for version 2 of the KeyNote trust-management system [KEYNOTE]. X.509 certificates [RFC3280] can be directly used in the Authorizer or Licensees field (or in both fields) in a KeyNote assertion, allowing for easy integration with protocols that already use X.509 certificates for authentication.

In addition, the document defines additional signature types that use other hash functions (beyond the MD5 and SHA1 hash functions that are defined in [RFC2792]).

1. Introduction

KeyNote is a simple and flexible trust-management system designed to work well for a variety of large- and small-scale Internet-based applications. It provides a single, unified language for both local policies and credentials. KeyNote policies and credentials, called `assertions', contain predicates that describe the trusted actions permitted by the holders of specific public keys. KeyNote assertions are essentially small, highly-structured programs. A signed assertion, which can be sent over an untrusted network, is also called a `credential assertion'. Credential assertions, which also serve the role of certificates, have the same syntax as policy assertions but are also signed by the principal delegating the trust. Note that only one principal may sign a credential assertion, but trust may be delegated to multiple principals. The credential assertion may delegate trust to each of these principals separately, or to groups of principals required to act together. For more details on KeyNote, see [KEYNOTE]. This document assumes reader familiarity with the KeyNote system.

Cryptographic keys may be used in KeyNote to identify principals. To facilitate interoperation between different implementations and to allow for maximal flexibility, keys must be converted to a normalized canonical form (depended on the public key algorithm used) for the purposes of any internal comparisons between keys. For example, an RSA key may be encoded in base64 ASCII in one credential, and in hexadecimal ASCII in another. A KeyNote implementation must internally convert the two encodings to a normalized form that allows for comparison between them. Furthermore, the internal structure of an encoded key must be known for an implementation to correctly decode it. RFC 2792 [RFC2792] describes the RSA and DSA key identifier and signature encodings for use in KeyNote assertions. This document specifies a new key identifier, allowing X.509 certificates [RFC3280] to be used as a key substitute wherever an RSA or DSA key may be used in KeyNote. Specifically, KeyNote will use the key associated with the Subject of an X.509 certificate. In addition, this document defines a corresponding signature encoding, to be used in conjunction with X.509 key identifiers. Finally, this document defines new signature encodings that use new hash functions beyond the MD5 and SHA1 functions defined in RFC 2792, and which in recent years have been found to be vulnerable to attack.

1.1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in RFC 2119 [RFC2119].

2. X.509 Key Identifier Encoding

X.509 key identifiers in KeyNote are encoded as an ASN1 DER encoding of the whole X.509 certificate, as defined in Section 4 of RFC 3280 [RFC3280].

For use in KeyNote credentials, the ASN1 DER-encoded object is then ASCII- encoded (e.g., as a string of hex digits or base64 characters).

X.509 keys encoded in this way in KeyNote must be identified by the "x509-XXX:" algorithm name, where XXX is an ASCII encoding ("hex" or "base64"). Other ASCII encoding schemes may be defined in the future.

3. X.509 Key Identifier Normalized Forms

For comparison purposes, X.509 certificates are parsed, the Subject public key is extracted and decomposed according to the rules described in Section 2 of [RFC2792]. The resulting RSA or DSA key is then used for comparing, per [RFC2792]. All X.509 key comparisons in KeyNote occur between normalized forms. Note that this allows for comparison between a directly encoded RSA or DSA key (as specified in RFC 2792) and the same key when contained in an X.509 certificate.

4. X.509 Signature Computation and Encoding

X.509 key identifier signatures are defined for historical reasons. Implementers are encouraged to use the RSA or DSA-based signature encodings instead.

X.509 key identifier signatures in KeyNote are identical to RSA- or DSA-based signatures [RFC2792]. The only difference is that the public key corresponding to the private key that generated the signatures is encoded in an X.509 certificate in the ``Authorizer'' field of the signed credential assertion. However, an RSA- or DSA-based signature encoding (depending on the Subject key contained in the X.509 certificate itself) may be used instead.

X.509 key identifier signatures in KeyNote are computed over the assertion body (starting from the beginning of the first keyword, up to and including the newline character immediately before the "Signature:" keyword) and the signature algorithm name (including the trailing colon character, e.g., "sig-x509-sha512-base64:")

X.509 key identifier signatures are encoded as an ASN1 OCTET STRING object, containing the signature value.

For use in KeyNote credentials, the ASN1 OCTET STRING is then ASCII-encoded (as a string of hex digits or base64 characters).

X.509 key identifier signatures encoded in this way in KeyNote must be identified by the "sig-x509-XXX-YYY:" algorithm name, where XXX is a hash function name (see Section 5 and Section 7 of this document) and YYY is an ASCII encoding ("hex" or "base64").

5. Hash Functions For RSA, DSA, and X.509 Key Identifier Signatures

For historical reasons (backward compatibility), X.509 key identifier signatures SHOULD support SHA1 as the hash function, using the "sha1" keyword. In addition, SHA256, SHA512 and RIPEMD160 [SHA256+] [SHA2-2] [RIPEMD-160] signatures MUST be supported for use with X509 key identifier signatures, by using the "sha256", "sha512" and "ripemd160" keywords respectively (see Section 7).

In addition, SHA256, SHA512 and RIPEMD160 signature identifiers are defined for RSA signatures, using the "sha256", "sha512" and "ripemd160" keywords respectively (see Section 7).

6. Security Considerations

This document discusses the format of X.509 keys and signatures as used in KeyNote. The security of KeyNote credentials utilizing such keys and credentials is directly dependent on the strength of the related public key algorithms. On the security of KeyNote itself, see [KEYNOTE]. Furthermore, it is the responsibility of the application developer to ensure that X.509 certificates are valid (signed by a trusted authority, not expired, and not revoked).

The use of SHA1 as part of signatures and key identifiers is discouraged, because of the various weaknesses in the algorithm that have been identified in recent years.

7. IANA Considerations

Per [KEYNOTE], IANA should provide a registry of reserved algorithm identifiers. The following identifiers are reserved by this document as public key identifier encodings:

- "x509-hex"
- "x509-base64"

The following identifiers are reserved by this document as signature encodings:

- "sig-x509-sha1-hex"
- "sig-x509-sha1-base64"
- "sig-x509-sha256-hex"
- "sig-x509-sha256-base64"
- "sig-x509-sha512-hex"
- "sig-x509-sha512-base64"
- "sig-x509-ripemd160-hex"
- "sig-x509-ripemd160-base64"
- "sig-rsa-sha256-hex"
- "sig-rsa-sha256-base64"
- "sig-rsa-sha512-hex"
- "sig-rsa-sha512-base64"
- "sig-rsa-ripemd160-hex"
- "sig-rsa-ripemd160-base64"

Note that the double quotes are not part of the algorithm identifiers.

8. Normative References

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[SHA256+] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9. Informative References

[KEYNOTE] Blaze, M., Feigenbaum, J., Ioannidis, J., and A. Keromytis, "The KeyNote Trust-Management System, Version 2", RFC 2704, September 1999.

[RIPEMD-160] 3.ISO/IEC 10118-3:1998, "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions," International Organization for Standardization, Geneva, Switzerland, 1998.

[RFC2792] Blaze, M., Ioannidis, J., and A. Keromytis, "DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System", RFC 2792, March 2000.

[SHA2-2] NIST, "Descriptions of SHA-256, SHA-384, and SHA-512", May 2001, <http://csrc.nist.gov/cryptval/shs/sha256-384-512.pdf>.

10. Acknowledgements

The author would like to thank Jim Schaad for his review and comments on earlier versions of this document.

Authors' Addresses

Angelos D. Keromytis
Department of Computer Science
Columbia University
Mail Code 0401
1214 Amsterdam Avenue
New York, New York 1007
USA
angelos <at> cs <dot> columbia <dot> edu